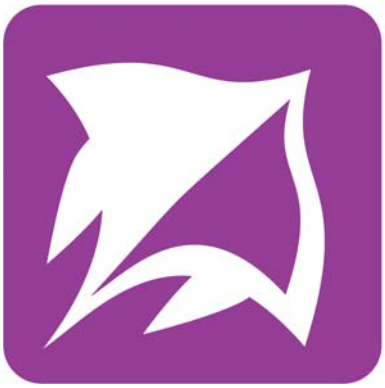


INM SecureNet Xtra

For Adobe Director
Version 2.0.0

User Manual

© Integration New Media Inc. 2002-2009| Manual Version 2.0.0 | 2009-08-06

Contents

Contents **2**

License agreement **5**

Introduction **7**

Before you read this manual 7

Easy to learn and use 7

Behaviors 7

Easy to deploy applications 7

Tutorials and Samples 8

Requirements and installation **9**

System requirements 9

Special considerations and limitations 9

Installation 9

How to register your SecureNet Xtra license 10

SecureNet Xtra basics **11**

What is a proxy server? 11

Proxy server settings 12

Defining proxy server settings in Internet Explorer 12

Defining proxy server settings on Mac OS 9**Error! Bookmark not defined.**

Defining proxy server settings on Mac OS X 13

Initializing the proxy server settings for SecureNet Xtra 14

Secure HTTP (HTTPS) outside browser 14

Basic network communication tasks 14

Naming conventions 15

Synchronous vs. asynchronous modes 15

When to use asynchronous mode 15

When to use synchronous mode 16

Formatting URLs 16

Using SecureNet Xtra Behaviors **18**

Before using behaviors 18

snbGetNetText 18

snbDownloadNetThing 19

Sending data with snbForm behaviors 19

snbForm.Field 19

snbForm.Submit 19

Using SecureNet Xtra Lingo methods **20**

Requesting data 20

Sending data 21
Monitoring the progress of requests 21
Handling Errors with Lingo 22
Display errors on screen 22
Trap errors without displaying 23
Troubleshooting network communications 24
Troubleshooting proxy server connections 24
Delivering to the end user 25
Standalone projectors 25
Shockwave projectors and movies 25
Delivering a Shockwave Projector 25
Delivering a Shockwave movie 25
Auto-installation 25
Location of Shockwave Xtra file 26
Where to go next 28
FAQs 28
Samples and tutorials 28
Latest user manual 28
Discussion list **Error! Bookmark not defined.**
Technical support 28
Appendix 1: SecureNet Xtra vs. NetLingo 29
Appendix 2: Methods reference 31
snxDownloadNetThing 31
snxGetNetText 33
snxGetStreamStatus 34
snxNetAbort 35
snxNetDone 35
snxNetError 36
snxNetTextResult 37
snxPostNetText 37
snxProxyServer 39
Appendix 3: Behaviors reference 41
Script 'Error Manager' 41
snsDisplayError 41
snsNetError 42
snbForm 43
snbForm.Field 43
snbForm.Submit 43
snbGetNetText 45
snbDownloadNetThing 47

Appendix 4: Error codes **49**
Index **59**

Apple, Mac and Macintosh are trademarks or registered trademarks of Apple Computer, Inc. in the United States and/or other countries. Adobe, Authorware, Director and Xtra are trademarks or registered trademarks of Adobe, Inc. in the United States and/or other countries. Microsoft, Internet Explorer, Windows and Windows NT are trademarks or registered trademarks of Microsoft Corporation, registered in the U.S. and/or other countries.

Other trademarks, trade names and product names contained in this manual may be the trademarks or registered trademarks of their respective owners, and are hereby acknowledged.

License agreement

PLEASE READ THIS LICENSE AGREEMENT CAREFULLY BEFORE USING Integration New Media's SecureNet XTRA. BY USING Integration New Media's SecureNet XTRA, YOU AGREE TO BECOME BOUND BY THE TERMS OF THIS LICENSE AGREEMENT.

The enclosed computer program(s), license file and data (collectively, "Software") are licensed, not sold, to you by Integration New Media, Inc. ("INM") for the purpose of using it for the development of your own products ("Products") only under the terms of this Agreement. INM and its licensors reserve any rights not expressly granted to you. INM and its licensors grant you no right, title or interest in or to the Software. The Software is owned by INM and its licensors and is protected by International copyright laws and international treaties.

1. **License**.

You may install one copy of the Software on a single Windows-compatible computer and one copy on a single Macintosh-compatible computer. To "install" the Software means that the Software is either loaded or installed on the permanent memory of a computer (i.e., hard disk). This installed copy of the Software may be accessible by multiple computers, however, the Software cannot be installed on more than one computer at any time. You may only install the Software on another computer if you first remove the Software from the computer on which it was previously installed. You may not sublease, rent, loan or lease the Software.

You may make one copy of the Software in machine-readable form solely for backup purposes. As an express condition of this Agreement, you must reproduce on each copy any copyright notice or other proprietary notice that is on the original copy supplied by INM.

Your license is limited to the particular version (collectively "Version") of the Software you have purchased. Therefore, use of a Version other than the one encompassed by this License Agreement requires a separate license.

Upon purchase of a Software license, you are given a license number, which is subject to the restrictions set forth above and may not be distributed by you in any way. However, INM and its licensors grant you a royalty-free right to reproduce and distribute the file named "SecureNet Xtra.x32" ("Runtime Kit") provided that (i) you distribute the Runtime Kit only in conjunction with and as part of your own Products; (ii) own a license for the specific Version of the Software that contains the Runtime Kit; (iii) agree to indemnify, hold harmless and defend INM and its licensors from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your Products with the Runtime Kit.

Any third party who will use the Runtime Kit in an authoring environment must purchase its own license of the Software.

Notwithstanding any other terms in this License, if the Software is licensed as an upgrade or update, then you may only use the Software to replace previously validly licensed versions of the same software. You agree that the upgrade or update does not constitute the granting of a second license to the Software (i.e., you may not use the upgrade or update in addition to the software it is replacing, nor may you transfer the software which is being replaced to a third party).

2. **Restrictions**.

The Software contains trade secrets in its human perceivable form and, to protect them, you may not MODIFY, TRANSLATE, REVERSE ENGINEER, REVERSE ASSEMBLE, DECOMPILE, DISASSEMBLE OR OTHERWISE REDUCE THE SOFTWARE TO ANY HUMAN PERCEIVABLE FORM. YOU MAY NOT MODIFY, ADAPT, TRANSLATE, RENT, LEASE, LOAN OR CREATE DERIVATIVE WORKS BASED UPON THE SOFTWARE OR ANY PART THEREOF.

THE SOFTWARE IS NOT INTENDED FOR USE IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, AIR TRAFFIC CONTROL, OR OTHER ENVIRONMENTS IN WHICH THE FAILURE OF THE SOFTWARE COULD LEAD TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

You may not transfer or assign your rights under this License to another party without INM's prior written consent. Assignment application forms can be obtained from INM's sales department.

3. **Copyright notices**. You may not alter or change INM's and its licensors' copyright notices as contained in the Software. You must include a copyright notice, in direct proximity to your own copyright notice, in substantially the following form "Portions of code are Copyright ©2002-2006 used under license by Integration New Media, Inc.".

4. **Acceptance**. The Software shall be deemed accepted by you upon delivery unless you provide INM, within two (2) weeks therein, with a written description of any bona fide defects in material or workmanship.

5. **Termination**. This Agreement is effective until terminated. This Agreement will terminate immediately without notice from INM or judicial resolution if you fail to comply with any provision of this Agreement. Upon such termination you must destroy the Software, all accompanying written materials and all copies thereof, and Sections 6 and 7 will survive any termination.

6. **Limited Warranty**. INM warrants for a period of ninety (90) days from your date of purchase (as evidenced by a copy of your receipt) that the media on which the Software is recorded will be free from defects in materials and workmanship under normal use and the Software will perform substantially in accordance with the manual. INM's entire liability and your sole and exclusive remedy for any breach of the foregoing limited warranty will be, at INM's option, replacement of the disk, refund of the purchase price or repair or replacement of the Software.

7. **Limitation of Remedies and Damages**. In no event will INM, its parent or subsidiaries or any of the licensers, directors, officers, employees or affiliates of any of the foregoing be liable to you for any consequential, incidental, indirect or special damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of business information and the like), whether foreseeable or not, arising out of the use of or inability to use the Software or accompanying written materials, regardless of the basis of the claim and even if INM or an INM representative has been advised of the possibility of such damage. INM's liability to you for direct damages for any cause whatsoever, and regardless of the form of the action, will be limited to the greater of US $199.00 or the money paid for the Software that caused the damages.

THIS LIMITATION WILL NOT APPLY IN CASE OF PERSONAL INJURY ONLY WHERE AND TO THE EXTENT THAT APPLICABLE LAW REQUIRES SUCH LIABILITY. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

8. **General**. This Agreement will be construed under the laws of the Province of Quebec, except for that body of law dealing with conflicts of law. If any provision of this Agreement shall be held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this Agreement will remain in full force and effect.

9. The parties acknowledge having requested and being satisfied that this Agreement and its accessories be drawn in English. Les parties reconnaissent avoir demandé que cette entente et ses documents connexes soient rédigés en anglais et s'en déclarent satisfaits.

Introduction

Integration New Media's SecureNet Xtra enables you to create Director projectors and Shockwave movies that can:

- Access Internet/Intranet resources even from behind any proxy server, whether or not username and password authentication is required;
- Access Internet/Intranet resources from secure web sites using the HTTPS protocol;

In addition to these unique features, SecureNet Xtra helps you reduce costs by:

- Enabling you to implement HTTP queries in a much easier way than NetLingo;
- Allowing you to debug HTTPS queries right from the Director authoring environment;

Before you read this manual

This user manual provides the essential information you need to use SecureNet Xtra. The section entitled SecureNet Xtra basics outlines the typical scenarios in which SecureNet Xtra is used.

Easy to learn and use

If you're already familiar with NetLingo, you already know how to use INM's SecureNet Xtra. SecureNet Xtra provides nearly all the methods found in NetLingo and the method names and syntaxes are almost identical. The only visible difference is the "snx" at the beginning of all SecureNet Xtra method names. This makes converting any existing NetLingo code to SecureNet Xtra a breeze.
SecureNet Xtra methods can be used in place of NetLingo methods without any additional coding, whether in a setting that requires encrypted authentication or not.

Behaviors

In addition to the methods available in the Xtra itself, INM has included in the download package, **SecureNet Xtra Behaviors** that allow you to quickly accomplish the basic tasks of sending and retrieving data over the Internet. These Behaviors are FREE for you to use and modify as needed.

Easy to deploy applications

Your standalone projectors can be delivered on CD-ROM, or downloaded onto the end-user's hard drive. The end-user doesn't need any special player. You don't even have to know your user's proxy server settings. At authoring time you can choose to allow SecureNet Xtra to detect the IP address and port number based on the user's current Internet settings. This also means the user's IT staff doesn't have to become involved in deploying your solutions. See System requirements for complete specifications.

Tutorials and Samples

In addition to the Xtra itself, INM provides a set of samples, tutorials and documentation to assist you.
The **First Steps** tutorial gives you step-by-step instructions on how to use SecureNet Xtra's Behaviors to complete a sample application that demonstrates sending credit card information over a secure connection.
With the **Proxy-Connect** tutorial you learn how to use SecureNet Xtra's Behaviors to connect from a computer behind a proxy server, to a World Wide Web server that returns data.
Both of these tutorials come with sample Director movies that you can use and modify for your own projects.
Download the First Steps and Proxy-Connect tutorials from our web site:
http://inm.com/products/snxdirector/samples/

Requirements and installation

System requirements

- Adobe Director version 11.0 – 11.5
- Windows XP/Vista
- Windows: Microsoft Internet Explorer 6 and later (or at least the corresponding wininet.dll library).
- Mac OSX 10.4.x – 10.5.x
- Mac: working Internet connection.
- Mac: URL Access Manager library version 2.4.1 or higher.

Special considerations and limitations

➢ **SQUID proxy limitations**

- With SQUID proxy on Mac, the POST operation is only supported for small amounts of data (less than 7Kb).
- With SQUID proxy on Windows, you may POST large amounts of data by using the workaround solution described in the Tech Note called "How to handle large POST requests through SQUID proxy server using snxPostNetText".

Note: Commonly used web browsers also have problems with SQUID; Safari does not support SQUID proxy, and Internet Explorer does not support large POST operations consistently with SQUID when user authentication is required.

Installation

1. Make sure Adobe Director is not running,
2. Place the Xtra (on Windows, a file named *"SecureNet Xtra.x32";* on Mac, a file named *"SecureNet Xtra.XTR"*)) into the Xtras folder. This folder is located in the same place as the Director application.
3. Place SecureNet Xtra's Behaviors Library (a file named *"SecureNet Xtra.cst"*) in Director's Libs folder.
4. Launch Director. To check if the Xtra was correctly installed type "showxlib" in the Message window. You should see "SecureNetXtra" in the list of Xtras displayed. To check if the Behaviors were correctly installed, open the Library Palette from the Windows menu. You should find the SecureNet Xtra Behaviors library from the pop-up menu in the upper left corner of the window.

How to register your SecureNet Xtra license

In order to ship a product containing SecureNet Xtra you must purchase a license for the software. Please make sure to read and agree to the License Agreement (see License agreement) before purchasing your SecureNet Xtra license. To purchase a license, visit our online store at:
https://store.INM.com/ ,

or contact us by phone at:
+1 800 400 1772 or +1 514 871 1333

> **Note:** The free evaluation version of SecureNet Xtra contains all the functionality of the licensed version, except that a Demo Version alert appears the first time the application calls a SecureNet Xtra method or uses a SecureNet Xtra Behavior. Click the alert to close it.

To register your Xtra, and remove the Demo splash screen, you need to call the SecureNet Xtra method `snxRegister(`*`thekey`*`)` at the beginning of each Director movie that uses the Xtra. The string parameter *theKey* is the license number that was given to you by INM upon purchase of the license, in quotes. To avoid making mistakes we recommend you copy/paste this number from the email that Integration New Media sends you.
Example: `snxRegister("S2R9-T6K8-2HD2-J391")`

> **Note:** Never distribute the license number. If your movie must be sent to other developers, first remove the license number from your script.

You would use SecureNet Xtra in these scenarios:

1) Your application crosses a proxy server that requires authentication
2) Your application requires a secure connection (HTTPS) from a projector or Shockwave movie.
3) Your application meets both of these requirements

What is a proxy server?

A proxy server intercepts requests between a client (e.g., your desktop) and a web server. Many corporate offices now use secure proxy servers for one or more of the following reasons:

- to mask the identity of the client computer; only the proxy server's IP address is seen from outside.
- to serve web pages from its cache and thus alleviate network traffic when the same URLs are requested repeatedly
- to control access to the web

Depending on a proxy server's security settings, it may require a username and password; this is known as authentication. As a multimedia developer, you must be aware that more and more potential clients are located in environments where they are behind proxy servers that may require username and password authentication before allowing any http communication. Therefore, if your application requires Internet access, you have to address this issue.

For example, corporate e-learning applications are often used by company members from both within the proxy and externally. If the e-learning application obtains content from outside the proxy, the internal users need SecureNet Xtra to be able to receive it. Likewise if the application sends tracking information to a database that is behind the proxy, SecureNet Xtra must be used to allow data to pass into the secure environment.

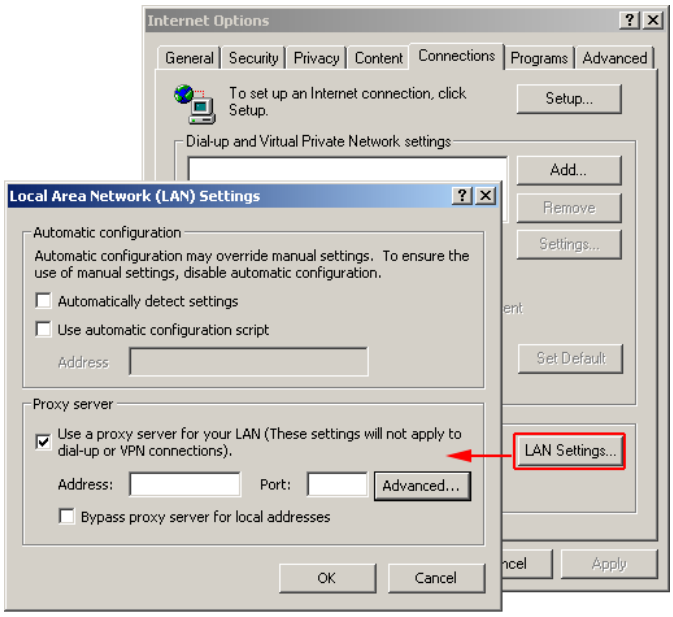
See the following definitions of "proxy server" for further clarification:
http://www.newworldpartnership.com/def_proxy.html
http://www.webopedia.com/TERM/p/proxy_server.html

Proxy server settings

On Windows, SecureNet Xtra uses Microsoft Internet Explorer's encryption schemes to send an encoded username and password that the proxy server is able to decode. On Mac, SecureNet Xtra uses the URL Access Manager library. On either platform, SecureNet Xtra can be set up to use the end-user's current Internet settings to obtain the proxy IP Address and Port number.

Defining proxy server settings in Internet Explorer

To set or check the proxy server settings used by Internet Explorer:

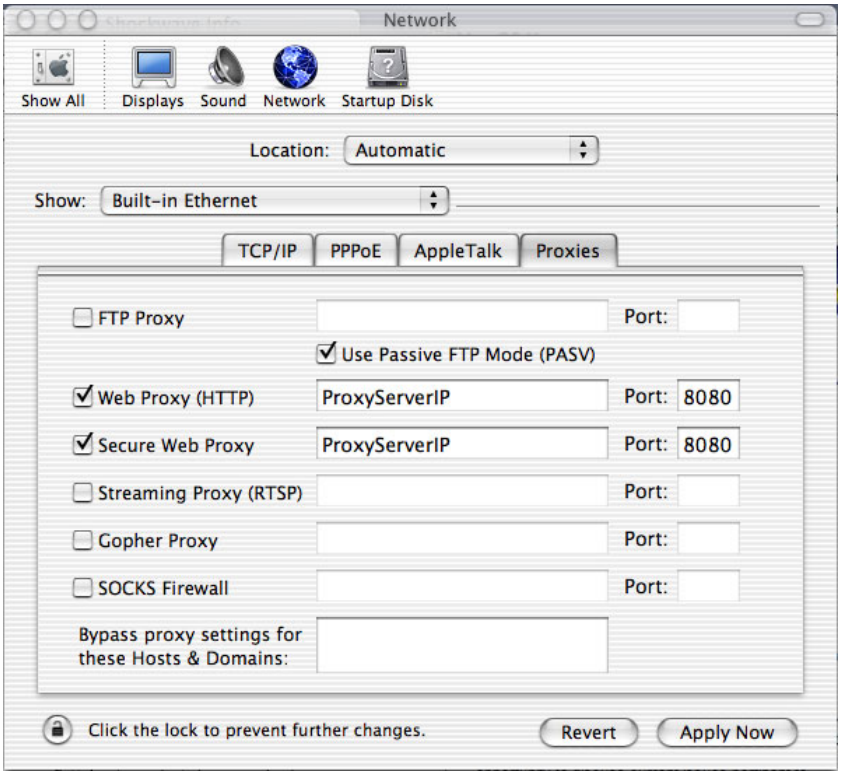
1. Click Tools > Internet Options… from the Internet Explorer menu bar.
2. In the Connections tab of the Internet Options dialog box, click LAN Settings…
3. The lower half of the LAN settings window contains Proxy server settings.
4. If you connect to the Internet using a proxy server, the checkbox should be checked and the Address and Port fields should be filled in with the correct settings.
5. Click Advanced… to view or set specifications for multiple proxy servers.
6. 

IE proxy server settings

Defining proxy server settings on Mac OS X

To change the Internet settings on Mac OS X:

1. Open **System Preferences...** from the Apple menu;
2. In the Internet & Network section, choose **Network;**
3. In the **Location:** and **Show:** menus, select the options that describe your Internet connection;
4. Click the **Proxies** tab;
5. Click **Web Proxy (HTTP)**, and enter the IP address or name of the proxy server and well as the port number;
6. To define a proxy server for secure HTTP operations, click **Secure Web Proxy**, and enter the IP address and port number of the proxy server.



Network preferences on Mac OS X: Proxy server settings

Note: On Mac OS X version 10.2, secure HTTP requests do not pass through the proxy server. If there is another way to access the Internet, the request will go directly through. See HTTPS across a proxy server with authentication.

On Mac OS X 10.3.x, secure HTTP communications do pass through the proxy server.

Initializing the proxy server settings for SecureNet Xtra

If you are using SecureNet Xtra Behaviors in your project, you specify the proxy server settings directly in the Behavior dialog boxes for each Behavior that performs a network communication task.
If you are coding your project in Lingo, you need to call the SecureNet Xtra method `snxProxyServer()`, with syntax 3 or 4, to set the proxy server information, before using any of the other SecureNet Xtra methods. This method lets SecureNet Xtra know that you will be using a proxy server, and identifies the proxy server *IP Address* and *Port number*, and the *Username* and *Password*, if required.
Examples:

```
snxProxyServer("204.19.171.45", 8080, "Vahe", "secretpwd")
snxProxyServer(#Default, "Vahe", "secretpwd")
```

Once `snxProxyServer()` is called to set the proxy server parameters, all subsequent SecureNet Xtra methods use these proxy server settings. When you want to stop using the proxy server, you can call `snxProxyServer` with the `#Stop` parameter, as in:

```
snxProxyServer(#Stop)
```

The snxProxyServer method can also be used to obtain information about the proxy server settings in use. For more details see snxProxyServer in Appendix 2: Methods reference.

Secure HTTP (HTTPS) outside browser

For this scenario, your end-user is not necessarily behind a proxy, but your application requires a secure connection, with encrypted data transfer, and your application must run outside the browser environment.
An example of this application is a kiosk that sells tickets to sporting events. It checks a remote database to see if tickets are available. It then accepts the buyer's name and credit card information, encrypts the information and sends it to another remote application that verifies it. If the information is valid, the kiosk prints the ticket using information that it receives, encrypted, from the remote database application.

Basic network communication tasks

The basic tasks that SecureNet Xtra allows you to do are:

- **Request** data (either text or encrypted files from a URL)
- **Send** data (either text or encrypted files to a URL)
- **Monitor** the status of the connection to a remote URL

To perform these tasks you can use either the **SecureNet Xtra Behaviors** or the **Lingo methods**. The SecureNet Xtra Behaviors make it easy to rapidly implement basic communication tasks without having to know Lingo or worry about error management. The Behaviors all operate in synchronous mode, meaning the Director application waits for network communication to be completed before allowing any further interaction from the user. Using the Lingo methods, you can execute communication tasks in synchronous or asynchronous modes. This allows you to have more control over the flow of your application. The following sections discuss the differences between synchronous and asynchronous communication and indicate when to use each. If you are already familiar with these concepts, you may skip these

sections and go directly to Using SecureNet Xtra Behaviors or Using SecureNet Xtra Lingo methods.

> **Note for Shockwave developers:**
> Do not use SecureNet Xtra Behaviors if you are developing an application for Shockwave on Mac OS 9.x. Please see Limitation for Shockwave on Mac OS 9.

Naming conventions

SecureNet Xtra methods and Behaviors have names that are similar to their corresponding NetLingo methods. The distinguishing differences are the following prefixes:

- SecureNet Xtra Lingo methods begin with "**snx**"
- SecureNet Xtra Behaviors begin with "**snb**"
- SecureNet Xtra Scripts (found in the Behaviors Library) begin with "**sns**"

Synchronous vs. asynchronous modes

SecureNet Xtra can be used in synchronous or asynchronous modes.
Whether you request or send data, communication with the remote server is initiated, the server does something, and then the server sends back some result.
In **synchronous mode**, the Director application waits to receive the results of that communication (either data that was requested, or an acknowledgement that the data sent was received by the remote server). While Director is waiting, it does not process any other commands and does not allow the user to interact with objects on the stage.
In **asynchronous mode**, the Director application does not wait to receive the response. The next line of code is executed immediately. In this case, the application can process other data or do other tasks while waiting for the result to be returned. In asynchronous mode, the application typically checks to see if results have been returned in an `on idle`, `on exitframe` or `on enterframe` script.
NetLingo only supports asynchronous communication.

When to use asynchronous mode

When the user has to wait for a network operation to complete and you want to display something or play a sound or animation while the request is being processed, you should use asynchronous communication.
A good example of asynchronous communication is an application that runs queries on a large database or on a slow server, or if the user has a slow connection. If the queries take a while to process, you don't want your application to stop running between each one. Instead, you can allow a user to create and send queries asynchronously. At any time, the user can display a list of all queries generated and see which ones have returned results. This list can be updated by an `on idle` script.

Note: It may be impractical to allow more than four or five network processes to be active at any time. Depending on the processor and the amount of information transferred with each process, performance may deteriorate significantly.

When to use synchronous mode

You may only use synchronous communication if your application is **not** running as a Shockwave movie on the Mac.
Synchronous communication is more intuitive for developers and for many end-users as well. In many situations, a user performs an action and expects results immediately. Depending on the amount of data transferred over the network, network speed, and how frequent network communication occurs in your application, you may decide that synchronous communication results in uncomfortably long waits. If this is the case, you should use asynchronous communication, which must be done via SecureNet Xtra's Lingo methods.

To implement synchronous communication you use the same SecureNet Xtra methods as with asynchronous mode, but you add a repeat loop after each method is called to prevent Director from continuing.

Synchronous mode example:

```
NetID = snxGetNetText("http://www.myURL.com")
-- this repeat loop prevents Director from doing anything
-- while the network operation is still in progress
repeat while NOT (snxNetDone(NetID))
end repeat
err = snxNetError(NetID)
if err = "OK" then
   member( "DisplayField").text =  snxNetTextResult(NetID)
else
   alert "Error: no data returned. The error code is: "& err
end if
```

Note: During synchronous communication, all animation and sound are stopped until the end of the repeat loop. This also means there is no way for the user to abort the communication.

Formatting URLs

Whether you use SecureNet Xtra methods or Behaviors to request or send data over the Internet, the URLs must be properly formatted, according to the following rules:

- They must start with "http://" or "https://".
- If username and password are required by the destination server, they must be included in the URL using this syntax:

```
"https://myUsername:myPassword@www.SecureBuy.com/Store.html"
```

- URLs must be encoded. They cannot contain non-ASCII characters, ASCII control characters, reserved characters, or unsafe characters. The following table lists the characters that must be encoded for SecureNet Xtra URLs, with their hexadecimal representations. To encode a character within a URL, replace it by the "%" symbol, followed by the two-digit hexadecimal representation.

Character	Hexadecimal representation
Reserved characters	
Forward slash/Virgule ("/")	%2F
Colon (":")	%3A
'At' symbol ("@")	%40
Unsafe characters	
Space	%20
Quotation marks	%22
Percent (“%”)	%25
Caret (“^”)	%5E
Tilde (“~”)	%7E
Left square bracket (“[”)	%5B
Right square bracket (“}”)	%5D
Grave accent (“`”)	%60

Using SecureNet Xtra Behaviors

The SecureNet Xtra Behaviors make it easy to rapidly implement basic communication tasks without having to know Lingo or worry about error management. The Behaviors all operate in synchronous mode, meaning the Director application waits for network communication to be completed before allowing any further interaction from the user.

> **Note for Shockwave developers:**
> Because the Behaviors (except for the ErrorManager scripts) execute in synchronous mode, they cannot be used if you are developing an application for Shockwave on Mac OS 9. Please see Limitation for Shockwave on Mac OS 9.

Before using behaviors

If you are using the SecureNet Xtra Behaviors, you don't have to worry about error checking. Just copy the Error Manager script to your each of your movies or into a castlib that is shared among your movies. The Behaviors use this script to display error messages to the end user.

snbGetNetText

The `snbGetNetText` Behavior allows you to get data from a http or https server, in the form of text. You can use this Behavior to perform queries on a remote database by accessing ASP or CGI scripts. This Behavior performs the same operation as the `snxGetNetText()` method.
To use `snbGetNetText`, drag it onto a button on stage that will initiate the request when it is clicked. The URL for the request must be stored in a cast member of type text or field.
Fill in the parameters dialog box:
URL stored in: Select the cast member (of type text or field) that contains the URL.
Result stored in: Select a cast member (of type text or field) in which to store the returned text.
Timeout: Drag the slider to set the timeout value in seconds.
If your application connects through a proxy server you can either enter the proxy server IP Address and Port number, or allow SecureNet Xtra to use the values from MS Internet Explorer. Keep the **Use default proxy settings** check box checked to use Internet Explorer's settings. If there is no proxy server, *Use default proxy settings* has no effect.
If you are not using MS Internet Explorer's settings, enter the proxy server's IP address and port number manually in the corresponding fields
If your end user has a proxy server that requires username and password authentication, select the cast members where the username and password are stored, respectively.

> **Important Note:** Once you assign a cast member to a SecureNet Behavior parameter, such as *URL stored in*, you cannot change the location of this member in the cast.
>
> If the member is moved, the Behavior will no longer work properly.

snbDownloadNetThing

This Behavior allows you to download a file from a http or https server and save it to a specified name and location. snbDownloadNetThing performs the same operation as the `snxDownloadNetThing()` method. Note that snbDownloadNetThing cannot be used in a Shockwave context.
To use `snbDownloadNetThing`, drag it onto a button on stage that will initiate the download when it is clicked. The URI (Uniform Resource Identifier) for the file to download must be stored in a cast member of type text or field. The parameters for this Behavior are nearly identical to those of `snbGetNetText`. The only exception is that instead of '*Results stored in'*, you need to provide a local filename and path to which the downloaded file will be saved. The path and filename should be stored in a Director cast member that can be either on or off stage. In the `snbDownloadNetThing` parameters dialog box, select that cast member for the *'Local file name in'*: parameter.

Sending data with snbForm behaviors

The Behaviors `snbForm.Field` and `snbForm.Submit` allow you to easily create a data entry form that can be submitted to a URL. These two Behaviors are used in combination: `snbForm.Field` identifies the values to be submitted and `snbForm.Submit` sends the data to the specified URL.

snbForm.Field

Drag this Behavior from the Library palette to a sprite of type text or field on Director's stage.
Enter the parameters in the dialog box:
Form Name: this parameter is used to group the fields that are sent when a submit button is clicked that contains the same Form Name reference. It can be any string and does not correspond to anything on the receiving URL.
Item Name: this parameter identifies the variable on the server side that will receive the string data contained within the sprite.

snbForm.Submit

Drag this Behavior from the Library palette to a button, or any sprite that can receive the mouseUp event, on Director's stage.
Enter the parameters in the dialog box:
Form Name: the value of this parameter should match that of all the sprites to be submitted to the URL when the submit button is clicked. Form Name can be any string; it does not correspond to anything on the receiving URL.
URL stored in: select the cast member that contains the URL
Result stored in: select a cast member in which to store the resulting text
Timeout: drag the slider to set the timeout value in seconds

The proxy server parameters for this Behavior are identical to those used in `snbGetNetText` and `snbDownloadNetThing`.

Using SecureNet Xtra Lingo methods

Using the SecureNet Xtra methods directly, rather than through Behaviors, allows you to have greater control over the flow of your application. One advantage to using the methods is that they can be implemented in synchronous or asynchronous modes. See Synchronous vs. asynchronous modes. Note that you must use the asynchronous mode if you are distributing your project via Shockwave that will be viewed on the Mac.

Requesting data

You initiate a request for data from a remote server using either `snxGetNetText()` or `snxDownloadNetThing()`. `SnxGetNetText()` allows you to specify the URL for a text file, ASP or CGI script or web page and obtains the result as text. `SnxDownloadNetThing()` lets you download a file that is either text or non-text, and save it to a specified location.
After initiating a request, your application calls `snxNetDone()`, with the ID of the request, to determine if the result has been returned.
When `snxNetDone` returns a value of TRUE, you must call `snxNetTextResult()` to obtain the text data returned by the remote server after initiating a request with `snxGetNetText()`. You can then display the result or use it in your code. . After the call to `snxNetTextResult()`, the network ID is no longer valid and your code should set its variable to 0.

Asynchronous example:

```
global gNetID
on mouseUp
   gNetID = snxGetNetText("http://www.SecureNetXtra.com")
   -- check for errors in initiating the request
   snsDisplayError(gNetID,1)
end

on exitFrame
   if (gNet > 0) then   -- make sure the ID is valid
      if snxNetDone(gNetID) then
        if NOT snsDisplayError(gNetID,1) then
            member("Result").text = snxNetTextResult(gNetID)
        end if
        gNetID = 0  -- reset the network ID
      end if
      -- loop on the frame until the operation is done
      go to the frame
   end if
end
```

Note: In asynchronous mode, you must use a global network ID, gNetID in this example, so that you can check its status from outside the handler that initiates the network operation.

When using `snxDownloadNetThing()`, you do not have to call `snxNetTextResult(gNetID)`. As soon as `snxNetDone(gNetID)` is TRUE, the file should be downloaded to the local machine.

> **Note:** `snxDownloadNetThing()` cannot be used in a Shockwave context; SecureNet Xtra will return error code -2101.

Sending data

`snxPostNetText()` allows you to send or "post" data to a HTTP or HTTPS server. For example, you would use snxPostNetText() to submit form data. As with `snxGetNetText()`, your application calls `snxNetDone()`, with the ID returned from the `snxPostNetText` method, to determine if a result has been returned from the remote URL.
When `snxNetDone` returns a value of TRUE, you must call `snxNetTextResult()` to obtain the text data returned by the remote server. You can then display the result or use it in your code. After the call to `snxNetTextResult()`, the network ID is no longer valid and your code should set its variable to 0. Even if the script that was targeted by the POST does not return anything, you should call snxNetTextResult to clean up the memory.

Asynchronous example:

```
global gNetID
on mouseUp
   gNetID = snxPostNetText (http://www.inm.com/orderform.cgi,
   [#color:"red", #quantity: "200"])
   -- check for errors in initiating the post
   snsDisplayError(gNetID,1)
end

on exitFrame
    if (gNetID > 0) then  -- make sure the ID is valid
        if snxNetDone(gNetID) then
             if NOT snsDisplayError(gNetID,1) then
                       member("Result").text = snxNetTextResult(gNetID)
             end if
             gNetID = 0  -- reset the network ID
        end if
        go to the frame
    end if
end
```

Monitoring the progress of requests

After initiating a request or sending data to a remote URL, you need to check to see that the request was received before you can display or do anything with result returned.
To find out when a network request has completed, call `snxNetDone(NetID).` It returns `True` if the request has completed (with or without errors), or `False` if the operation is still in progress.

`SnxGetStreamStatus(NetID)` allows you to monitor the status of any SecureNet Xtra network operation. It returns a propertyList containing the status of the network operation identified by `NetID`. The format of the property list is

([#URL:"xxx", #state: "yyyy", #bytesSoFar: z, #bytesTotal: u, #error: e]), where
#URL: the current URL
#state: current state. The value can be “Connecting”, “Started”, “InProgress”, “Complete” or “Error”.
#bytesSoFar : number of bytes transferred.
#bytesTotal : total number of bytes to transfer. This value may be equal to #bytesSoFar if the server doesn’t include the content length in the header.
#Error : Empty string if the download is not complete, "OK" if it completed successfully, or an error code (<0) if the operation ended with an error.

You can use the properties #bytesSoFar and #bytesTotal to display a progress bar to the user.

Handling Errors with Lingo

The SecureNet Xtra method `snxNetError(NetID)` simply returns an error code if an error occurred during the request specified by NetID. However, if you want the text description of the error, you can take advantage of the methods in the Error Manager script, found in the SecureNet Xtra Behaviors Library. Copy that script to your internal cast.
After each call to `snxGetNetText()`, `snxPostNetText()` or `snxDownloadNetThing()`, call either `snsNetError(`*`NetID`*`),`or `snsDisplayError(`*`NetID,detail`*`)` where *`NetID`* is the unique network operation ID returned by `snxGetNetText()`, `snxPostNetText()` or `snxDownloadNetThing()`.

> **Note:** The Error Manager script does not perform any communications; it just interprets the results of snxNetError. Therefore, you may use this script in any context, Mac or Windows.

Display errors on screen

`snsDisplayError()` displays the error message directly to the end-user. You have the option of displaying a short error description (detail = 0) or a more detailed error description with suggested possible causes (detail=1).

Example (using asynchronous mode):

```
global gNetID
on mouseup
   gNetID = snxDownloadNetThing(URL, fName)
   if (snsDisplayError(gNetID, 1)) then
    -- displays the error directly in an alert box
    exit
   end if
end mouseup

on exitFrame
   -- download or query was initiated successfully
   -- now check for completion
   if (gNetID > 0) then -- make sure the ID is valid
    if snxNetDone(gNetID) then
        -- request completed, check the result status
```

```
        if (snsDisplayError(gNetID, 1)) then
                -- error; do something else
        else
        --  the file is downloaded successfully
        -- open the file
        end if
    gNetID = 0  -- reset the network ID
    else
    -- update a status bar on screen and continue to wait
    member("txtSTATUS").text = string(snxGetStreamStatus(gNetID))
   go to the frame
end if
   end if
end exitFrame
```

Trap errors without displaying

`snxNetError()` returns an error code (<0) if an error occurred. `snsNetError()` simply returns a list containing the short error description and the more detailed description. You may call these methods to trap and manage errors yourself.

Example (using asynchronous mode):

```
global gNetD
on mouseup
   gNetID = snxDownloadNetThing(URL, fName)
   if (snxNetError(gNetID) <> "OK") then
    lMyError = snsNetError(gNetID)
    -- do something, because you have an error.
    -- lMyError contains the description and
    -- details of the error
    exit
   end if
end mouseup

on exitFrame
   -- download or query was initiated successfully
   -- now check for completion
   if (gNetID > 0) then
    if snxNetDone(gNetID) then
        -- request completed, check the result status
        if (snxNetError(gNetID) <> "OK") then
                -- do your own error handling
                gNetID = 0
                exit
        end if
        --  the file is downloaded successfully
        gNetID = 0
    else
    -- update a status bar on screen
        go to the frame -- loop
    end if
   end if
end exitFrame
```

See Script 'Error Manager' in the Behaviors reference for more details on using these methods.

Troubleshooting network communications

Network communications may fail due to any of several possible conditions:

- internal communication problems in the user's network
- syntax error in the URL or in the parameters sent via a post
- invalid local file name for snxDownloadNetThing
- server dead
- proxy server not working or wrong username and password
- wrong or missing username and password in the URL for an https request

This section outlines how to test where the problem lies.

Troubleshooting proxy server connections

The steps in this flowchart should be performed from within the proxy server network, usually on the end-user's computer. Before you begin, make sure you have the correct Proxy Address and Port number settings in Internet Explorer. See Defining proxy server settings in Internet Explorer .

Step 1. INM provides a HTML file named testproxy.htm with the SecureNet Xtra package. Copy this file to the server your application connects to. Then type the URL of the HTML file (the one on your server) directly in Internet Explorer. Example:

`http://myserver/mytest/testproxy.htm`

Does it work?

Yes: If the URL returns a message saying the "connection is working" it means there is no error in getting past the proxy server and the error must lie at the Director end, as opposed to within the network. Continue to Step 2.

No: If the result of Step 1 was not successful – you did not receive any feedback-- it means the problem lies within the internal network configuration or the proxy settings in Internet Explorer. Contact the System Administrator at the user's site.

Step 2. Play the Proxy-Connect sample Shockwave movie available from: http://www.SecureNetXtra.com/demos. Replace the URL with the URL you used in Step 1.

Does it work?

Yes: If this sample works, it means the problem lies within your Director scripts or server-side scripts. Continue to Step 3.

No: If this sample does not work, there may be an error in SecureNet Xtra. Contact INM's support team at:
http://www.INM.com/support/

Step 3. Try debugging your code. Verify that the URL you are trying to access is properly encoded. See Formatting URLs.

If you need assistance, Integration New Media's Developer Assistance program can give you detailed support on your projects. Visit our Developer Assistance page on our web site to find out more about this service.
http://www.INM.com/services/developer-assistance/

Delivering to the end user

SecureNet Xtra is designed in a way that minimizes any last minute changes needed before delivery to the end-user. (For security and usability reasons, Adobe has developed a different process for locating and installing, over the Internet, for Shockwave applications – See Shockwave projectors and movies below).

Standalone projectors

You deliver the Xtra file (on Windows: **SecureNet Xtra.X32**, on Mac: **SecureNet Xtra.XTR**) in a folder named "Xtras" located in the same folder as the projector itself.

Shockwave projectors and movies

Delivering a Shockwave Projector

A Shockwave Projector is very similar to a Director Projector, except that the file you deliver is very tiny; it contains no Xtras and no Director playback engine.
See Auto-installation below for information on how to make sure that the appropriate version of SecureNet Xtra will be downloaded to the user's computer when needed.
To create a Shockwave Projector, select File > Create Projector > Options > Player: Shockwave.

> **Note:** If you plan to distribute your Shockwave projector or movie for cross-platform usage, be sure to fully test it on both Mac and Windows platforms. See Special considerations and limitations.

Delivering a Shockwave movie

SecureNet Xtra cannot be included in Shockwave movies. Instead, it is automatically downloaded to the end-user's computer, if required (i.e., if not already installed).
See Auto-installation below for information on how to make sure that the appropriate version of SecureNet Xtra is downloaded to the user's computer when needed.
To test the resulting Shockwave movie locally on your computer, create a folder named DSWMEDIA anywhere on your hard disk and move the deliverable files into it (i.e., the Shockwave movie, the external castlibs, the HTML file, etc.)

Auto-installation

When you deliver your project as a Shockwave movie, SecureNet Xtra can be automatically installed onto your end-users computers via the Internet. To instruct your movie to auto-install SecureNet Xtra when needed:

1. Make sure (or add) the Verisign information in the "xtrainfo.txt" file located in your Director folder.

Important: This information is different for each release of the Xtra. There fore you should **copy it directly from the release notes** that come with the SecureNet Xtra download package. Everything in square brackets must be on a single line without line breaks. It looks something like this:

```
; SecureNet Xtra for Director by Integration New Media, Inc.
; http://www.INM.com/
[#namePPC:"SecureNet Xtra.XTR",#nameW32:"SecureNet
   Xtra.X32",#package:"http://signed.INM.com/verisign/snxdirector/1.3.2/Se
   cureNetXtra",#info:"http://www.INM.com/products/snxdirector"]
```

2. Launch Director and open your movie.
3. Choose Modify > Movie > Xtras.
4. If "SecureNet Xtra" is not in the list, click **Add**. Choose "SecureNet Xtra.X32" on Windows; or "SecureNet Xtra.XTR" on Mac, and click **OK**.
5. Select SecureNet Xtra in the list again and make sure "**Download if Needed**" is checked.
6. Now, **Save** your movie, and **Publish** the Shockwave movie.

Note: You should copy the exact text from the Xtra's Release Notes and paste it into your "Xtrainfo.txt" file, as this information changes with every release.

Users who access Shockwave movies that automatically download SecureNet Xtra will get a Verisign certificate with the Xtra's credentials. Verisign is a standard means of downloading software from secure sources. Users have the choice of allowing or refusing the installation of the Xtras.
The appropriate Xtras for their operating system will be downloaded into a standard location on the hard drive and will then be accessible to any Shockwave movie thereafter.

Note: If the user refuses to download the Xtras, your application will not work correctly. You may want to use some defensive programming techniques to recover from such a situation.

Location of Shockwave Xtra file

The location of the auto-downloaded Shockwave Xtra file is as follows:

- **Windows Vista**

C:\Users\User\AppData\LocalLow\Adobe\Shockwave Player 11\xtras\download\IntegrationNewMediaInc\INMSecureNetXtra> SecureNet Xtra.x32

- **Windows XP**

WINNT > System32 > Adobe > Shockwave 11 > Xtras > Download > IntegrationNewMediaInc > INMsSecureNetXtra > SecureNet Xtra.x32

- **Mac OS X**

Mac OS X/Users/Library/Application Support/Adobe/Shockwave /Xtras/download/IntegrationNewMediaInc/INMsSecureNetXtra/SecureNet Xtra.XTR

Where to go next

FAQs

If you experience a problem with SecureNet Xtra, before you call Technical Support, please check the FAQs (Frequently Asked Questions) on our Web site to see if your problem is discussed there, and visit the Download section (http://inm.com/products/snxdirector/download/), to make sure you are using the most up-to-date version of SecureNet Xtra.

Samples and tutorials

INM provides sample Director movies and tutorials to help you get started quickly with SecureNet Xtra. Check the demos page on our web site at http://www.inm.com/products/snxdirector/samples/.

Latest user manual

Also, remember to read any relevant sections of the SecureNet Xtra User Manual. The user manual may be updated between releases of the product, to clarify or correct any misleading information. The latest version of the manual is available for download from INM's Download Library (http://www.inm.com/products/snxdirector/support/).

Technical support

INM offers free basic technical support to license owners as well as clients who are in the process of evaluating our products. However, **priority is given to license owners**. Make sure to provide your license number when you contact us to maintain your priority status.
To obtain support please submit a ticket on our web site:
http://support.inm.com/

Appendix 1: SecureNet Xtra vs. NetLingo

NetLingo methods	SecureNet Xtra methods and Behaviors	Differences with SecureNet Xtra
Downloading and streaming media		
DownloadNetThing	snxDownloadNetThing snbDownloadNetThing	• Only HTTP and HTTPS; not FTP • Allows downloading files from a secure HTTPS server from a projector or Director movie
GetNetText	snxGetNetText snbGetNetText	• Only HTTP and HTTPS; not FTP • Does not support multiple OS character set • Use of netID parameter is mandatory • Allows retrieval of text from a secure HTTPS server from a projector or Director movie
PostNetText	snxPostNetText snbForm.Field snbForm.Submit	• Only HTTP and HTTPS; not FTP • Does not support multiple OS character set • Use of netID parameter is mandatory • Allows posting to a secure HTTPS server from a projector or Director movie
Using Network Operations		
GetLatestNetID	NS-F	
GetStreamStatus	snxGetStreamStatus	• Does not support syntax with an URL passed as a parameter, only netID
NetAbort	snxNetAbort	• Does not support syntax with an URL passed as a parameter, only netID
NetDone	snxNetDone	• Use of netID parameter is mandatory
NetError	snxNetError snsNetError snsDisplayError	• snxNetError doesn't support the no parameter syntax • snsDisplayError and snsNetError allow for multiple syntaxes
NetLastModDate	NS-F	
NetMime	NS-F	
NetPresent	NS-F	
NetTextResult	snxNetTextResult snbGetNetText snbPostNetText	• Use of netID parameter is mandatory • The Behaviors allow you to optionally display the returned result directly to a field or text

		member.
On streamStatus	NS-F	
ProxyServer	snxProxyServer	SecureNet Xtra allows for additional syntaxes, with the ability to accept username and password
TellStreamStatus	NS-F	

NS-F: Not supported in this release; may be supported in future.

Appendix 2: Methods reference

snxDownloadNetThing

Syntax snxDownloadNetThing (*URL, localFile*)

Parameters *URL* is the URL of the file to download.
localFile is the full name (including the path) of the file on the local disk.

Return Success: unique id for the operation (> 0)
Failure: error code (<0)

Description Initiates the loading of a file on an HTTP or HTTPS server.
The URL must be properly encoded and must start with “http://” or “https://”.
To access a file on a HTTPS server, depending on how the server is configured, you may need to specify a user name and password within the URL, using the standard URL syntax.
Use snxNetDone to find out when the snxDownloadNetThing operation is complete. Use snxNetError (or the ErrorManager scripts in the SecureNet Xtra Behavior Library) to find out if the operation was successful.
Use snxGetStreamStatus to get the bytes downloaded so far.
Shockwave doesn’t support snxDownloadNetThing.

Examples

```
-- Asynchronous sample
global gNetID
on mouseUp
    gNetID = snxDownloadNetThing( "http://www.INM.com/webmedia/logo.gif",
    the applicationPath & "logo.gif")
end

global gNetID
on exitFrame
-- check if the operation is completed
    if (gNetID > 0) then
       if snxNetDone(gNetID) then
        if Not snsDisplayError(gNetID) then -- display any errors
           member("logo").fileName = the applicationPath & "logo.gif")
           go to frame label(“AfterResults”)
        end if
       else  -- update the status on screen and wait
        member("status").text = string(snxGetStreamStatus (gNetID))
        go to the frame
       end if
    end if
end

-- synchronous sample
on mouseUp
    NetID = snxDownloadNetThing( "http://www.inm.com/webmedia/logo.gif" ,
    the applicationPath & "logo.gif")
    Repeat while NOT (snxNetDone(NetID))
        member("status").text = string(snxGetStreamStatus (NetID))
```

```
    end repeat
    member("logo").fileName = the applicationPath & "logo.gif")
end
-- with username and password (accessing https server)
on mouseUp
    NetID = snxDownloadNetThing(
    "https://myUsername:myPassword@www.inm.com/webmedia/logo.gif", the
    applicationPath & "logo.gif")
    Repeat while NOT (snxNetDone(NetID))
        member("status").text = string(snxGetStreamStatus (NetID))
    end repeat
    member("logo").fileName = the applicationPath & "logo.gif")
end
```

> **Note:** In asynchronous mode, you must use a global network ID, gNetID in this example, so that you can check its status from outside the handler that initiates the network operation.

See also snxNetAbort, snxNetDone, snxNetError, snxGetStreamStatus

snxGetNetText

Syntax

Syntax1: `snxGetNetText (`*`URL`*`)`
Syntax2: `snxGetNetText(`*`URL, propertylist`*`)`

Parameters

URL is the URL to retrieve.

PropertyList is property list of parameters for CGI scripts.

Return

Success: unique id for the operation (> 0)
Failure: error code (<0)

Description

Initiates the loading of text from a file on an HTTP or HTTPS server, or initiates a CGI query.

The URL must be properly formatted and must start with “http://” or “https://”.
If you use syntax 2, parameters in *propertyList* don’t have to be URL encoded.
To access a file on a HTTPS server, depending on how the server is configured, you may need to specify a user name and password within the URL, using the standard URL syntax.
The length of the request (number of characters for the URL, plus parameters) is limited to the maximum allowed by Internet Explorer.
Use `snxNetDone` to find out when the `snxGetNetText` operation is complete. Use snxNetError (or the ErrorManager scripts in the SecureNet Xtra Behavior Library) to find out if the operation was successful. Use `snxNetTextResult` to return the text downloaded by `snxGetNetText`. Use `snxGetStreamStatus` to get the bytes downloaded so far.

Examples

```
-- Asynchronous sample
global gNetID
on mouseUp
    gNetID = snxGetNetText ("http://www.inm.com")
end

global gNetID
on exitFrame
    if (gNetID > 0) then -- make sure the ID is valid
        if snxNetDone(gNetID) then
            member("Result").text = snxNetTextResult(gNetID)
            gNetID = 0 -- reset the request ID
        end if
        go to the frame
    end if
end
-- synchronous sample
-- accessing HTTPS with username and password
-- required for destination server (NOT the proxy server)
on mouseUp
    NetID = snxGetNetText ("https://eric:INM@www.inm.com")
    Repeat while NOT (snxNetDone(NetID))
        member("status").text = string(snxGetStreamStatus (NetID))
```

```
    end repeat
    member("Result").text = snxNetTextResult(NetID)
end
-- Syntax 1: passing parameters to an ASP script
gNetID = snxGetNetText("http://www.inm.com/query.asp?name=eric%20Cartman")
-- Syntax 2: passing parameters to an ASP script
gNetID = snxGetNetText("http://www.integration.qc.ca/query.asp", [#name:
    “eric cartman"])
```

> **Note:** In asynchronous mode, you must use a global network ID, gNetID in this example, so that you can check its status from outside the handler that initiates the network operation.

See also snxNetAbort, snxNetDone, snxNetError, snxNetTextResult, snxGetStreamStatus

snxGetStreamStatus

Syntax `snxGetStreamStatus (`*`NetID`*`)`

Parameters *NetID* is the unique network operation ID returned by the snxGetNetText or snxDownloadNetThing method.

Return Success: propertyList that contains the status of the network operation ([#URL:"xxx", #state: "yyyy", #bytesSoFar: z, #bytesTotal: u, #error: e])

#URL: the current URL

#state: current state. The value can be Connecting, Started, InProgress, Complete or Error.

#bytesSoFar : number of bytes transferred.

#bytesTotal : total number of bytes to transfer. This value may be equal to #bytesSoFar if the server doesn’t include the content length in the header.

#Error : Empty string if the download is not complete, "OK" if it completed successfully, or an error code (<0) if the operation ended with an error.

Failure: error code (<0)

Description Reads the status of the network operation identified by *NetID*.

Examples

```
-- Asynchronous sample
global gNetID
on mouseUp
    gNetID = snxGetNetText ("http://www.inm.com")
end
on exitFrame
    if (gNetID > 0) then
        put snxGetStreamStatus(gNetID)
        if snxNetDone(gNetID) then
            member("Result").text = snxNetTextResult(gNetID)
            gNetID = 0
        end if
        go to the frame
    end if
```

```
end
```

It will display this kind of information in the message window:

```
-- [#URL: "http://www.inm.com", #state: "InProgress", #bytesSoFar: 3501,
    #bytesTotal: 50000, #error: ""]
```

> **Note:** In asynchronous mode, you must use a global network ID, gNetID in this example, so that you can check its status from outside the handler that initiates the network operation.

See also snxNetAbort, snxNetDone, snxNetError, snxNetTextResult, snxGetNetText, snxDownloadNetThing

snxNetAbort

Syntax `snxNetAbort (NetID)`

Parameters *NetID* is the unique network operation ID returned by snxGetNetText, snxPostNetText or snxDownloadNetThing method.

Return Nothing.

Description Cancel a network operation (initiated with snxGetNetText, snxPostNetText or snxDownloadNetThing) identified by *NetID*. It has no effect if the operation was already complete.

Examples

```
on mouseUp
    if (gNetID > 0) then
        snxNetAbort(gNetID)
    end if
end
```

See also snxGetNetText, snxPostNetText, snxDownloadNetThing

snxNetDone

Syntax snxNetDone (*NetID*)

Parameters *NetID* is the unique network operation ID returned by snxGetNetText, snxPostNetText or snxDownloadNetThing methods.

Return TRUE if the network operation is completed with success or with an error; FALSE if the network operation is still in progress.

Description Indicates if a network operation (initiated with snxGetNetText, snxPostNetText or snxDownloadNetThing) identified by *NetID* is finished.

Examples

```
global gNetID
on exitFrame
if (gNetID > 0) then
    if snxNetDone(gNetID) then
        if snxNetError(gNetID) <> "OK" then
            -- use the Error Manager script to display the error
            snsDisplayError(gNetID,1)
        else
            -- display the results from the server
```

```
            member("Result").text = snxNetTextResult(gNetID)
            go to frame label(”AfterResults”)
        end if
        gNetID = 0 -- reset the Network ID
    else
        go to the frame  -- continue to wait
    end if
end
```

See also snxNetTextResult, snxGetNetText, snxPostNetText, snxDownloadNetThing

snxNetError

Syntax snxNetError (*NetID*)

Parameters *NetID* is the unique network operation ID returned by snxGetNetText, snxPostNetText or snxDownloadNetThing method.

Return "OK" if the network operation completed with success.

An error code (<0) if the network operation didn't complete with success.

Empty string if the network operation is in progress or if *NetID* doesn't refer to a valid network operation.

Description Determines if an error occurred in a network operation

Examples

```
global gNetID
on exitFrame
    if (gNetID > 0) then   -- make sure the Network ID is valid
        if snxNetDone(gNetID) then
            if snxNetError(gNetID) <> "OK" then
                -- display the error code
                Alert("Error:"&&string(snxNetError(gNetID)))
                -- do something else
            else
                -- display the result text
                member("Result").text = snxNetTextResult(gNetID)
            end if
            gNetID = 0   -- reset the Network ID
        end if
    end if
    go to the frame
end
```

See also snxNetDone, snxNetTextResult, snxGetNetText, snxPostNetText, snxDownloadNetThing

snxNetTextResult

Syntax snxNetTextResult (*NetID*)

Parameters *NetID* is the unique network operation ID returned by snxGetNetText or snxPostNetText method.

Return If successful: The text retrieved from the URL
Otherwise: Empty string

Description Retrieves the text read during the network operation identified by *NetID*. After this method is called, the network ID no longer exists.

Examples

```
global gNetID
on exitFrame
    if (gNetID > 0) then -- make sure the Network ID is valid
        if snxNetDone(gNetID) then
        -- use ErrorManager script to display text of the error
            if NOT snsDisplayError(gNetID,1) then
                -- no error; display the result text
                member("Result").text = snxNetTextResult(gNetID)
            end if
            gNetID = 0 -- reset the Network ID
        end if
    end if
    go to the frame
end
```

See also snxNetDone, snxNetError, snxNetTextResult, snxGetNetText, snxPostNetText

snxPostNetText

Syntax Syntax1: snxPostNetText (*URL, postText*)
Syntax2: snxPostNetText (*URL, propertylist*)

Parameters *URL* is the URL to retrieve.
PostText is the text to post.
PropertyList is property list of parameters to post.

Return Success: unique id for the operation (> 0)
Failure: error code (<0)

Description Initiates the posting of parameters to a CGI script on an HTTP or HTTPS server. The information is sent in the same way a browser posts an HTML form (using METHOD=POST).
The URL must be properly formatted and must start with “http://” or “https://”.
To access a file on an HTTPS server, you must specify a user name and password using the standard URL syntax.
When using syntax 1 (passing parameters in a string), the parameters are sent to the server using MIME type "text/plain" and are not URL-encoded.
When using syntax 2 (passing parameters in a property list), the parameters are sent to the server using MIME type "application/x-www-form-urlencoded" and are URL-encoded.
Unlike with `snxGetNetText()`, there is no limitation on the number of

characters in the request.
Use snxNetDone to find out when the snxPostNetText operation is complete. Use snxNetError (or the ErrorManager scripts in the SecureNet Xtra Behavior Library) to find out if the operation was successful. Use snxNetTextResult to return the text returned by snxPostNetText. Use snxGetStreamStatus to get the bytes downloaded so far.

Examples

```
-- Asynchronous sample
global gNetID
on mouseUp
    -- initiate the POST
    gNetID = snxPostNetText (http://www.inm.com/orderform.cgi,
    [#color:"red", #quantity: "200"])
end

global gNetID
on exitFrame
    if (gNetID > 0) then -- make sure the Network ID is valid
        if snxNetDone(gNetID) then
            -- display the results and continue
            member("Result").text = snxNetTextResult(gNetID)
            gNetID = 0 -- reset the Network ID
            go to frame label(“AfterResults”)
        else   -- update the status info and keep waiting
        member("status").text=string(snxGetStreamStatus(NetID))
            go to the frame
        end if
    end if
end

-- synchronous sample (accessing HTTPS)
-- Username and password required for
-- the destination server, NOT the proxy server
on mouseUp
    NetID = snxPostNetText ("https://Vahe:INM@www.inm.com/orderform.cgi",
    [#color:"red", #quantity: "200"])
    Repeat while NOT(snxNetDone(NetID))
        member("status").text = string(snxGetStreamStatus(NetID))
    end repeat
    member("Result").text = snxNetTextResult(NetID)
end
```

Note: In asynchronous mode, you must use a global network ID, gNetID in this example, so that you can check its status from outside the handler that initiates the network operation.

See also snxNetAbort, snxNetDone, snxNetError, snxNetTextResult

snxProxyServer

Syntax

Syntax1: snxProxyServer ()
Syntax2: snxProxyServer (*Command*)
Syntax3: snxProxyServer (*IPAddress*[, *Port*[, *UserName*[, *Password*]]])
Syntax4: snxProxyServer (*#Default,[UserName[,Password]]*)

Parameters

Syntax1:

Reads the current proxy server settings into a propertyList ([#Address: "xxx", #Port: yyy, #UserName: "zzz", #Password: "uuu"])
On Windows, if snxProxyServer was set using Syntax 4 (#Default), this method checks for proxy server settings according to the following algorithm:

1) If an auto-config script is used, snxProxyServer() returns "AutoConfigURL".
2) If 'use same proxy for all protocols' is checked, it returns the HTTP proxy settings
3) Else if HTTP settings are not empty, it returns them
4) Else if Secure HTTP (https) settings are not empty, it returns them
5) Else if Socks settings are not empty, it returns them

Note: Syntax 1 does not retrieve the #UserName and #Password properties unless they were explicitly set using syntax 3 or 4 below.

Syntax2:

command is a symbol that indicates the action to be performed by snxProxyServer:
#Stop: stop using a proxy server for network operations
#Address: returns the address of the proxy server
#Port: returns the port used by the proxy
#UserName: returns the user name used by the proxy
#Password: returns the password used by the proxy

Syntax3:

IPAddress is the IP address or URL of the proxy server (String)
Port is the port number used by the proxy server (Integer). This value can be omitted (the default value is 80)
UserName is the user name to use with a proxy server that requires authentication. If omitted, the default value is "".
Password is the password to use with a proxy server that requires authentication. If omitted, the default value is "".

Syntax4:

#Default: Sets the proxy settings to Microsoft Internet Explorer's settings on Windows, and to the Internet Control Panel's settings on Mac.
UserName is the user name to use with a proxy server that requires authentication. If omitted, the default value is "".
Password is the password to use with a proxy server that requires

authentication. If omitted, the default value is "".

Return Depends on the syntax used (see Parameters and Description).

Description Syntax1: Reads the proxy server settings into a propertyList ([#Address: "xxx", #Port: yyy, #UserName: "zzz", #Password: "uuu"])
If there is no proxy server, the property list returned is empty.
Syntax2: Executes an action (see Parameters for details)
Syntax3: Sets the proxy server configuration
Syntax4: Sets the proxy server configuration to use the system's default proxy server settings (IE settings on Windows; System Network Preferences on Mac OS X).

Examples

```
--syntax 3 to set the proxy server
-- with specific IP address and port number
snxProxyServer("204.19.171.45", 8080, "Vahe", "INM")
--syntax 3 with URL of the proxy server
snxProxyServer("http://myDomain.com/myProxy", 8080, "Vahe", "INM")

--syntax 4 to set the proxy server with default Internet settings
snxProxyServer(#Default, "Vahe", "INM")

--syntax 1 to read the proxy server settings
Put snxProxyServer()
-- ([#Address: "204.19.171.45", #Port: 8080, #UserName: "Vahe", #Password: "INM"]

-- syntax 2 to read the proxy server IP address
Put snxProxyServer(#Address)
-- "204.19.171.45"
```

Appendix 3: Behaviors reference

SecureNet Xtra Behaviors allow you to perform three basic network operations: get text, download a file, and submit data to a URL. If you installed the Behaviors properly, you should see the library named SecureNet-Xtra in Director's library palette. See

Installation if you need to review this procedure.
Before using a Behavior, drag the Error Manager script from the Behavior library to your internal cast. The Behaviors use this script to handle errors. To use a Behavior, drag it from the Library palette to the appropriate sprite on Director's stage and fill in the parameters. The Behavior will automatically be copied to your internal cast.
All SecureNet Xtra Behaviors are executed in synchronous mode, meaning the application waits for a response from the network operation before continuing. Each Behavior allows you to set a *Timeout* parameter that is used to abort the operation if no results have been returned within the specified timeout delay.

Script 'Error Manager'

This is a movie script that contains two methods used for managing error messages generated by SecureNet Xtra methods. If you use SecureNet Xtra Lingo methods, you can copy the methods in this script to your movie script and call these methods to check for errors. See Handling Errors with Lingo.
If you are using the SecureNet Xtra Behaviors, copy the Error Manager script from the Behavior library into your movie's internal cast. The Behaviors use this script to handle errors.

> **Note**: In order for the Behaviors to generate error descriptions you must copy the Error Manager script from the Behavior library to your internal cast.

snsDisplayError

Syntax snsDisplayError (*theError*, [*theDetailLevel*])

Parameters *theError* is the error code returned by a SecureNet Xtra method. Possible values:
`theError < 0`:displays the error message and returns TRUE
`theError = "OK" or ""`: displays nothing and returns FALSE
`theError = propertyList returned by snxGetStreanmStatus`: checks the error code, displays the error (if there is one), returns TRUE if there was an error and FALSE otherwise
`theError > 0` (assumes it's a valid NetId and not an error code); in this case `snsDisplayError` calls `snxNetError`, displays the error (if there is one), returns TRUE if there was an error and FALSE otherwise

TheDetailLevel is an optional parameter, possible values:
`0`: only the error description is displayed
`1`: the error description, plus suggested causes are displayed

Return TRUE if there was an error, FALSE if no error

Description Displays an alert containing the error description for error number `theError`, generated by a SecureNet Xtra method. Call this method after each SecureNet Xtra method to display the error message to the end-user. See parameters for more details.

Examples

```
NetID = snxGetNetText(
    http://www.SecureNetXtra.com/proxyconnect.html,[#user:"vahe", #password:
    "INM"])
-- Display the full text description of the error
snsDisplayError(NetID,1)
```

See also snsNetError, snxNetError, snxGetStreamStatus

snsNetError

Syntax snsNetError (*theError*)

Parameters *TheError* is the error code returned by a SecureNet Xtra method.
Possible values:
`theError < 0`:displays the error message and returns TRUE
`theError = "OK" or ""`: displays nothing an returns FALSE
`theError = propertyList returned by snxGetStreanmStatus`: checks the error code, returns the property list if there was an error and FALSE otherwise
`theError > 0` (assumes it's a valid a NetID and not an error code); in this case `snsNetError` calls the method `snxNetError()`, returns the property list if there is an error and FALSE otherwise

Return If `theError` is not valid, snsNetError returns an empty list.
If `theError` is a valid error number, snsNetError returns a property list: ([#Desc: '*Short description*', #Detail : '*possible causes*']).

Description Returns a property list containing the error description and details for error number `theError`, generated by a SecureNet Xtra method. Call this method when you want to trap and manage the error yourself.

Examples

```
NetID = snxGetNetText(
    http://www.SecureNetXtra.com/proxyconnect.html,[#user:"vahe", #password:
    "INM"])
err = snxNetError(NetID)
If err <> "OK" then
    lMyError = snsNetError(NetID)
```

See also snsDisplayError, snxNetError, snxGetStreamStatus

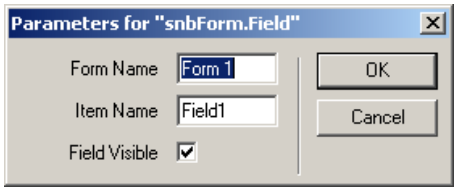
snbForm

There are two Behaviors: snbForm.Field and snbForm.Submit that are used in combination to create a form that can be submitted to a URL

snbForm.Field

This Behavior can only be attached to sprites of type 'field' or 'text'. It identifies the variables that each sprite value corresponds to on the server-side form. It must be used in combination with the snbForm.Submit Behavior, which posts all values with the same *Form Name* to the URL.

- **Interface**



- **Parameters**

Form Name: string containing the name of the form to which you are submitting data. The default for this parameter is "Form 1".
Item Name: string containing the name of the form variable that is recognized by the server.

Example : if `Item Name` is 'amountpaid', the Behavior executes this corresponding snxPostNetText method:

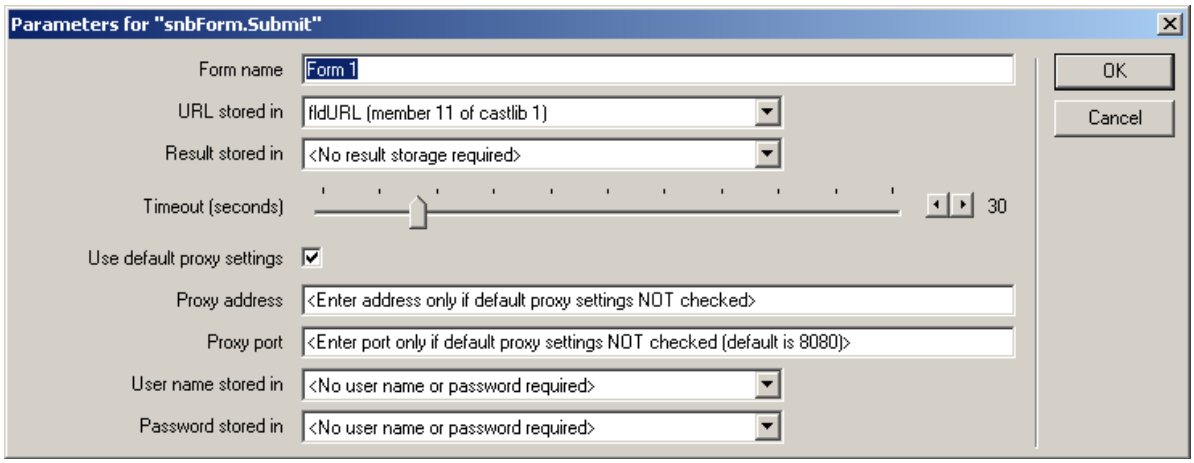
```
snxPostNetText("http://www.INM.com/orderform.cgi", [#amountpaid : 150]).
```

Field Visible: If checked, the sprite is visible at runtime; otherwise, the sprite is not visible. The default value is "checked".

snbForm.Submit

This Behavior may be used on any sprite that receives the "on mouseup" event. It must be used in combination with 'snbForm.Field'. When the sprite's mouseup event is triggered, the data identified by corresponding snbForm.Field Behaviors is sent to the specified URL.

➢ Interface



➢ Parameters

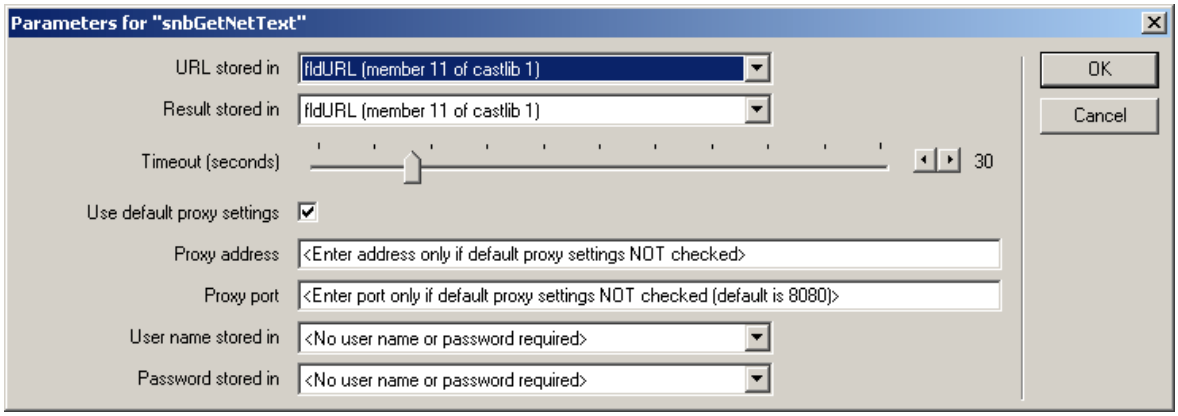
'Form Name': string containing the name of the form to which you are submitting data. The form name groups the fields to be sent to the server. The default value is "Form 1".
'URL stored in': the cast member that contains the URL of the form to which you are submitting data. The popup list contains all the available members of type Field or Text.
'Result stored in': the member in which the results of the request are stored. The popup list contains all the available members of type Field or Text. The default value for this property is the first member of type Field or Text available.
'Timeout': the delay in seconds to wait for a response from the URL before considering the communication a failure. The default value for this parameter is 30.
The remaining parameters are used only if a proxy server is used. If you are not using a proxy server, leave the default parameters.
'Use default proxy settings': If checked, the Behavior will use the proxy server settings obtained from MS Internet Explorer, and the parameters specified in 'Proxy Address', 'Proxy Port' are ignored. This parameter is "checked" by default.
'Proxy Address': character string containing the IP address or domain name of the proxy server. The default value for this parameter is empty.
'Proxy Port': character string containing the port number used by the proxy server. The default value for this parameter is 8080.
'Username stored in': the cast member that contains the Username for the proxy server. The popup list contains all the available members of type Field or Text. The default value for this property is empty.
'Password stored in': the cast member that contains the Password for the proxy server. The popup list contains all the available members of type Field or Text. The default value for this property is empty.

> **Note:** If the application does not use a proxy server, leave Use default proxy settings checked.
>
> If **Use default proxy settings** is not checked then you must have a proxy server and enter the correct IP Address and port number, or else these text boxes must be blank.

snbGetNetText

This Behavior can be attached to a button or any sprite type that receives the "on mouseup" event.
You must identify the cast member (type Text or Field) that contains the URL of the request and the member (type Text or Field) into which the result of the request will be stored.
The request is initiated when the "on mouseup" message is sent to the sprite.

➢ Interface



➢ Parameters

'URL stored in': the cast member that contains the URL of the text file or script for the request. The popup list contains all the available members of type Field or Text.
'Result stored in': the member to which the results of the request are stored. The popup list contains all the available members of type Field or Text. The default value for this property is the first member of type Field or Text available.
'*Timeout'*: the delay in seconds to wait for a response from the URL before aborting the operation. The default value for this property is 30.

The remaining parameters are used only if a proxy server is used. If you are not using a proxy server, leave the default parameters.
'Use default proxy settings': If checked, the Behavior will use the proxy server settings obtained from MS Internet Explorer, and the parameters specified in 'Proxy Address', 'Proxy Port' are ignored. This parameter is "checked" by default.
'Proxy Address': character string containing the IP address or domain name of the proxy server. The default value for this parameter is empty.
'Proxy Port': character string containing the port number used by the proxy server. The default value for this parameter is 8080.

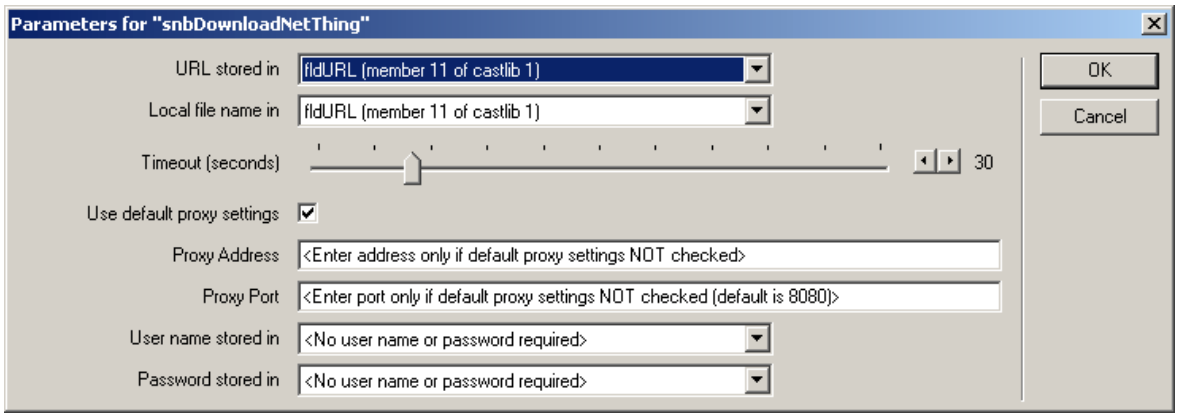
'Username stored in': the cast member that contains the Username for the proxy server. The popup list contains all the available members of type Field or Text. The default value for this property is empty.
'Password stored in': the cast member that contains the Password for the proxy server. The popup list contains all the available members of type Field or Text. The default value for this property is empty.

> **Note**: If the application does not use a proxy server, leave **Use default proxy settings** checked.
>
> If Use default proxy settings is not checked then you must have a proxy server and enter the correct IP Address and port number, or else these text boxes must be blank.

snbDownloadNetThing

This Behavior is attached to a button or any sprite that receives the " on mouseup" event.
You must identify the cast member (type Text or Field) in which the URL for the download request is stored and the member (type Text or Field) that contains the full path and file name to which the downloaded file will be saved.
The request is initiated when the "on mouseup" message is sent to the sprite. snbDownloadNetThing cannot be used in a Shockwave context; SecureNet Xtra will return an error.

- **Interface**



- **Parameters**

'URL stored in': the name of the cast member that contains the URL of the file to be downloaded. The popup list contains all the available members of type Field or Text.
'Local file name in': the member that contains the path and name of the file to which the Behavior saves the downloaded file. The popup contains the list of all the members of type Field and Text available. The default value for this property is the first member of type Field or Text available.
'*Timeout'*: the delay in seconds to wait for a response from the URL before aborting the operation. The default value for this property is 30.

The remaining parameters are used only if a proxy server is used. If you are not using a proxy server, leave the default parameters.
'Use default proxy settings': If checked, the Behavior will use the proxy server settings obtained from MS Internet Explorer, and the parameters specified in 'Proxy Address', 'Proxy Port' are ignored. This parameter is "checked" by default.
'Proxy Address': character string containing the IP address or domain name of the proxy server. The default value for this parameter is empty.
'Proxy Port': character string containing the port number used by the proxy server. The default value for this parameter is 8080.
'Username stored in': the cast member that contains the Username for the proxy server. The popup list contains all the available members of type Field or Text. The default value for this property is empty.

'Password stored in': the cast member that contains the Password for the proxy server. The popup list contains all the available members of type Field or Text. The default value for this property is empty.

> **Note:** If the application does not use a proxy server, leave **Use default proxy settings** checked.
>
> If Use default proxy settings is not checked then you must have a proxy server and enter the correct IP Address and port number, or else these text boxes must be blank

Appendix 4: Error codes

-400:	HTTP error: Invalid syntax. The request could not be processed by the server due to invalid syntax.
-401:	HTTP error: Access denied. The requested resource requires user authentication. User name and/or password may be absent or incorrect in the URL
-403:	HTTP error: Request forbidden. The server understood the request, but is refusing to fulfill it.
-404	HTTP error: Object not found. The server has not found anything matching the requested URI (Uniform Resource Identifier).
-405	HTTP error: Method is not allowed. The method used is not allowed.
-406	HTTP error: No response acceptable to client found. The server didn't find any response acceptable to the client.
-407	HTTP error: Proxy authentication required. You must provide a valid user name and password to the proxy server using the snxProxyServer method.
-408	Server timed out waiting for request. HTTP error: The server didn't receive the request. Try again.
-409	HTTP error: User should resubmit with more information. The request could not be completed due to a conflict with the current state of the resource. You should resubmit with more information.
-410	HTTP error: The resource is no longer available. The requested resource is no longer available on the server, and no forwarding address is known.
-411	HTTP error: Content length required. The server refuses to accept the request without a defined content length.
-412	HTTP error: Failed to evaluate precondition in the request.

	The precondition given in one or more of the request header fields evaluated to false when it was tested on the server.
-413	HTTP error: Request entity too large.
	The server is refusing to process a request because the request entity is larger than the server is willing or able to process.
-414	HTTP error: Request URI too long.
	The server is refusing to service the request because the request URI (Uniform Resource Identifier) is longer than the server is willing to interpret.
-415	HTTP error: Unsupported media type.
	The server refuses to service the request because the request is in a format not supported by the requested resource for the requested method.
-500	HTTP error: Internal server error.
	The server encountered an unexpected condition that prevented it from fulfilling the request.
-501	HTTP error: Request not supported.
	The server does not support the functionality required to fulfill the request.
-502	Error response received from gateway.
	HTTP error: The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.
-503	HTTP error: Temporarily overloaded.
	The server is temporarily overloaded.
-504	HTTP error: Timed out waiting for gateway.
	The request was timed out waiting for a gateway.
-505	HTTP error: HTTP version not supported.
	The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.
-965	Socket library error: Resource temporarily unavailable.
	This error is reported by the Socket library. The operation cannot be completed immediately. It should be retried later.
-1001	Internet library error: Out of handles.
	This error is reported by the Internet library. No more handles could be generated at this time.
-1002	Internet library error: Timeout.

This error is reported by the Internet library. The request has timed out.

-1003 Internet library error: Extended error.

This error is reported by the Internet library. An extended error was returned from the server. This is typically a string or buffer containing a verbose error message.

-1004 Internet library error: Internal error.

This error is reported by the Internet library. An internal error has occurred.

-1005 Internet library error: Invalid URL.

This error is reported by the Internet library. The URL is not properly formatted. Here are some examples: http://www.INM.com https://user:password@www.INM.com

-1006 Internet library error: Unrecognized scheme.

This error is reported by the Internet library. The URL scheme could not be recognized or is not supported.

-1007 Internet library error: Server name not resolved.

This error is reported by the Internet library. The server name could not be resolved.

-1008 Internet library error: Protocol not found.

This error is reported by the Internet library. The requested protocol could not be located.

-1009 Internet library error: Invalid option.

This error is reported by the Internet library. A request to InternetQueryOption or InternetSetOption specified an invalid option value.

-1010 Internet library error: Bad option length.

This error is reported by the Internet library. The length of an option supplied to InternetQueryOption or InternetSetOption is incorrect for the type of option specified.

-1011 Internet library error: Option cannot be set.

This error is reported by the Internet library. The request option cannot be set, only queried.

-1012 Internet library error: Shutdown.

This error is reported by the Internet library. The Internet function support is being shut down or unloaded.

-1013 Internet library error: Incorrect user name.

This error is reported by the Internet library. The request to

connect and log on to a FTP server could not be completed because the supplied user name is incorrect.

-1014 Internet library error: Incorrect password.

This error is reported by the Internet library. The request to connect and log on to a FTP server could not be completed because the supplied password is incorrect.

-1015 Internet library error: Login failure.

This error is reported by the Internet library. The request to connect to and log on to a FTP server failed.

-1016 Internet library error: Invalid operation.

This error is reported by the Internet library. The requested operation is invalid.

-1017 Internet library error: Operation canceled.

This error is reported by the Internet library. The operation was canceled, usually because the handle on which the request was operating was closed before the operation completed.

-1018 Internet library error: Incorrect handle type.

This error is reported by the Internet library. The type of handle supplied is incorrect for this operation.

-1019 Internet library error: Incorrect handle state.

This error is reported by the Internet library. The requested operation cannot be carried out because the handle supplied is not in the correct state.

-1020 Internet library error: Not a proxy request.

This error is reported by the Internet library. The request cannot be made via a proxy.

-1021 Internet library error: Registry value not found.

This error is reported by the Internet library. A required registry value could not be located.

-1022 Internet library error:

Windows: Bad registry value.

Mac: Bad Configuration in Control Panels > TCP/IP.

This error is reported by the Internet library.

Windows: A required registry value was located but is an incorrect type or has an invalid value.

Mac: Make sure your TCP/IP settings are correct or see your system administrator.

-1023 Internet library error: No direct access.

	This error is reported by the Internet library. Direct network access cannot be made at this time.
-1024	Internet library error: No context.
	This error is reported by the Internet library. An asynchronous request could not be made because a zero context value was supplied.
-1025	Internet library error: No callback function.
	This error is reported by the Internet library. An asynchronous request could not be made because a callback function has not been set.
-1026	Internet library error: Request pending.
	This error is reported by the Internet library. The required operation could not be completed because one or more requests are pending.
-1027	Internet library error: Incorrect format.
	This error is reported by the Internet library. The format of the request is invalid.
-1028	Internet library error: Item not found.
	This error is reported by the Internet library. The requested item could not be located.
-1029	Internet library error: Cannot connect.
	This error is reported by the Internet library. The attempt to connect to the server failed. Could be a wrong port number, or server is busy.
-1030	Internet library error: Connection aborted.
	This error is reported by the Internet library. The connection with the server has been terminated. Could be host is down or network is down.
-1031	Internet library error: Connection reset.
	This error is reported by the Internet library. The connection with the server has been reset.
-1032	Internet library error: Force retry.
	This error is reported by the Internet library. Calls for the Internet function to redo the request.
-1033	Internet library error: Invalid proxy request.
	This error is reported by the Internet library. The request to the proxy was invalid.
-1036	Internet library error: Handle already exists.

This error is reported by the Internet library. The request failed because the handle already exists.

-1037 Internet library error: Security certificate date expired.

This error is reported by the Internet library. SSL certificate date that was received from the server is bad. The certificate is expired.

-1038 Internet library error: Security certificate common name invalid.

This error is reported by the Internet library. SSL certificate common name (host name field) is incorrect. For example, if you entered www.server.com and the common name on the certificate says www.different.com.

-1039 Internet library error: HTTP to HTTPS on redirect.

This error is reported by the Internet library. The application is moving from a non-SSL to a SSL connection because of a redirect.

-1040 Internet library error: HTTPS to HTTP on redirect.

This error is reported by the Internet library. The application is moving from a SSL to a non-SSL connection because of a redirect.

-1041 Internet library error: Mixed security.

This error is reported by the Internet library. Indicates that the content is not entirely secure. Some of the content being viewed may have come from unsecured servers.

-1042 Internet library error: Change post is non secure.

This error is reported by the Internet library. The application is posting and attempting to change multiple lines of text on a server that is not secure.

-1043 Internet library error: Post is non-secure.

This error is reported by the Internet library. The application is posting data to a server that is not secure.

-1044 Internet library error: The certificate on the server has expired.

This error is reported by the Internet library. URL cannot be accessed without a valid certificate.

-1045 Internet library error: SSL Protocol not supported by URL Access.

This error is reported by the Internet library. If you see this error, you may be able to fix it by trying your request again. If it doesn't work, try adding a certificate to your server that's

supported by URL Access. If that doesn't help, there is currently no other way to workaround this error. You should not see this error when running on Mac OS X 10.2 and later. (Mac only)

-1046 Internet library error: The server requires 128-bit encryption, but URL Access doesn't support it.

This error is reported by the Internet library. URL Access began supporting 128-bit encryption in version 2.1, which shipped with Mac OS 9.1

-1047 Internet library error: File empty error.

This error is reported by the Internet library. Indicates a failed data transfer operation.

-1150 Internet library HTTP Error: Header not found.

This error is reported by the Internet library. The requested header could not be located.

-1151 Internet library HTTP Error: Downlevel server.

This error is reported by the Internet library. The server did not return any headers.

-1152 Internet library HTTP Error: Invalid server response.

This error is reported by the Internet library. The server response could not be parsed.

-1153 Internet library HTTP Error: Invalid header

This error is reported by the Internet library. The supplied header is invalid.

-1154 Internet library HTTP Error: Invalid query request.

This error is reported by the Internet library. The request made to HttpQueryInfo is invalid.

-1155 Internet library HTTP Error: Header already exists.

This error is reported by the Internet library. The header could not be added because it already exists.

-1156 Internet library HTTP Error: Redirect failed.

This error is reported by the Internet library. The redirection failed because either the scheme changed (for example, HTTP to FTP) or all attempts made to redirect failed (default is five attempts).

-1201 Internet library error.

On Windows: Failed to load Internet Explorer library (wininet.dll).

On Mac: URL Access failed to load or the URL Access library is unavailable.

	Windows: Make sure Internet Explorer 4.x or greater is installed on the computer and that the computer has an active Internet connection. Mac: Check that URL Access is in the Control Panels > Extensions Manager.
-1202	Internet Explorer library error: Failed to set the callback procedure. Severe error that can be caused by an old or invalid version of wininet.dll.
-1203	Internet library error: Failed to authenticate the user. The user name and/or the password defined with the snxProxyServer are not accepted by the proxy server.
-1204	Internet library error: Failed to close the request. Failure attempting to close the request handle. System error. Close the application and restart to release this handle.
-1205	Internet Config. error: failed to start or to find the config file. Check if there is “Internet” in the Control Panels and if you can set properties.
-1352	Server Exception: Routing exception. Server is unable to forward the request to the appropriate target. See your system administrator.
-1353	Server Exception: Relationship exception. The trust relationship between this workstation and the primary domain failed. See your system administrator.
-1354	Server Exception: Gateway exception. Default Gateway is not properly configured. See your system administrator.
-1355	Server Exception: IP Address exception. IP Address is not properly configured. See your system administrator.
-1356	Server Exception: Server exception. Server is not properly configured. See your system administrator.
-1357	Server Exception: Invalid status exception. Invalid status reported by server.
-1358	Server Exception: Configuration exception. Server configuration can’t be loaded.

-1359	Server Exception: Socket exception. Cannot make client socket non-blocking.
-2001	Internal error: Failed to allocate memory. The system is running out of memory. Try to close some applications or restart your computer.
-2002	Internal error: Invalid port number. The proxy port number must be an integer greater than or equal to 0 and less than or equal to 65537.
-2003	Internal error: Failed to generate a temporary file name. The system is not able to generate a unique temporary file name because too many temp files are in use. Close the application and restart.
-2004	Internal error: Failed to create a temporary file. The working directory of the application may be on a read-only disk.
-2005	Internal error: Failed to open the temp file. The temp file may have been deleted or moved manually from its location. Close the application and restart.
-2006	Internal error: Failed to read/write to the temp file. The temp file may have been deleted or moved manually from its location. Close the application and restart.
-2007	Internal error: Invalid network operation ID. The network operation ID doesn't exist. Make sure you are using a valid ID and that you have not disabled this ID (for example, by calling snxNetTextResult).
-2008	Internal error: No URL specified. You must provide a valid URL.
-2009	Internal error: Internal failure. Severe error; may be caused by a memory problem. Restart the computer.
-2010	Internal error: Unsupported protocol. INM's SecureNet only supports HTTP and HTTPS protocols. Make sure that the URL starts with http:// or https://.
-2011	Normal Operation: Request aborted by user. User purposefully aborted the network operation.
-2012	Syntax Error: Invalid file name. Local destination file name for downloaded file is not a valid file

	name.
-2101	Wrong mode for this Xtra. This error is triggered when snxDownloadNetThing is used in Shockwave.
-2102	Not supported on Mac OS 10.1.x. This error is generated when calling snxProxyServer() to initialize a proxy server on Mac OS 10.1.x.
-2103	Not supported on Mac OS 10.1.x. when using a proxy. This error is generated when trying to perform a HTTP or secure HTTP operation across a proxy server on Mac OS 10.1.x.

Index

A

Asynchronous communication, 17
Auto-installation, 28

B

Basic tasks, 16
Behaviors
  snbDownloadNetThing, 21, 50
  snbForm, 46
  snbForm.Field, 21, 46
  snbForm.Submit, 21, 46
  snbGetNetText, 20, 48
  snsDisplayError, 44
  snsNetError, 45
  Using Behaviors, 20
Behaviors reference (Appendix 3), 44

D

Delivering to end user, 28
Demos, 8, 31
Discussion list, 31
Downloading files
  using Behaviors, 21
  using Lingo, 22

E

Errors
  Debugging, 24
  Error codes (Appendix 4), 52
Example
  Posting data to a form, 23
  Requesting data using Get, 22

F

FAQs, 31

I

Installation, 10
Internet Explorer
  proxy server settings, 13

L

Licensing, 10
Limitations, 9
Location of Shockwave Xtra file, 29

M

Mac limitations, 9
Mailing list, 31
Methods
  snxDownloadNetThing, 34
  snxGetNetText, 22, 36
  snxGetStreamStatus, 23, 37
  snxNetAbort, 38
  snxNetDone, 23, 38
  snxNetError, 39
  snxNetTextResult, 40
  snxPostNetText, 23, 40
  snxProxyServer, 42
Methods reference (Appendix 2), 34

N

NetLingo comparison chart (Appendix 1), 32

P

Posting data
  using Behaviors, 21
  using Lingo, 23
Proxy server
  definition, 12
Proxy server settings, 13
Proxy settings
  Internet Explorer, 13
  Macintosh, 14, 15

R

Registration, 10
Requesting data
  using Behaviors, 20
  using Lingo, 22
Requirements
  Director, 9
  Internet Explorer, 9

S

Shockwave, 28
  Delivering a Stand-alone, 28
  on Mac, 9
Standalone projectors, 28
Synchronous communication, 18

T

Technical support, 31
Troubleshooting
  Code, 24
  Network connections, 27

Tutorials, 8, 31

U

URLs
Encoding, 19
Formatting, 19